# Test Plan for Unity-1.0.0 Fuel Plugin

Test Plan for Unity-1.0.0 Fuel Plugin 0

Revision history 2

Unity Cinder Plugin 3

Developer’s specification 3

[TODO] 3

Limitations 3

Test strategy 3

Acceptance criteria 3

Test environment, infrastructure and tools 3

Product compatibility matrix 4

Type of testing 4

1. Installation testing 4

2. Functional testing 4

3. System testing 5

1. Install plugin and deploy environment 5

2. Modifying env with enabled plugin (removing/adding controller nodes) 6

3. Modifying env with enabled plugin (removing/adding compute node) 7

4. Uninstall of plugin with deployed environment 7

5. Uninstall of plugin 8

# Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Revision date** | **Editor** | **Comment** |
| 1.0 | 19.10.2016 | Peter Wang <peter.wang13@emc.com> | Unity Cinder 1.0.0 Test plan |

# Unity Cinder Plugin

*The Unity Fuel plugin provides a graceful way to deploy Unity as a storage back-end in Fuel environment. User can specify configuration options using Fuel web UI to get rid of the manual configuration.*

## Developer’s specification

*[TODO]*

## Limitations

*The Unity Fuel Plugin does not support deploying more than one Unity in one Fuel environment.*

**Prerequisites**

* *Unity storage is up and running.*
* *Pools are created for OpenStack use.*
* *Data path connection is set up:*
  + *For iSCSI: iSCSI interface is configured on Unity, make sure it’s accessible from both compute nodes and controller nodes*
  + *For FC: FC zoning should be done between compute nodes and controller nodes*

# Test strategy

The Unity Fuel plugin simplify the deployment of Unity cinder driver by leveraging the Fuel infrastructure. In general, plugin installs Unity cinder driver, changes configuration file (/etc/cinder/cinder.conf and /etc/nova/nova.conf) for openstack. Additionally, it installs

Multipath and iSCSI initiator tools if not present.

The test strategy is to ensure that all options in the GUI are properly processed and desired configuration file for related components (cinder, nova) are generated.

## Acceptance criteria

*To validate that the Unity plugin is correctly deployed, configuration files need to be checked, related service state should be checked and basic volume operations should be performed without any error.*

## Test environment, infrastructure and tools

*The test environment shall include 6 VMs within a KVM or Virtualbox based environment. Following is an example of designations for the VMs:*

1. Fuel master node (w/ 50GB Disk, 2 Network interfaces [Mgmt, PXE] )
2. OpenStack Controller #1 node
3. OpenStack Controller #2 node
4. OpenStack Compute node (60GB disk)
5. OpenStack Cinder(CEPH-OSD) node1 (100GB disk)
6. OpenStack Cinder(CEPH-OSD) node2 (100GB disk)

Each VM shall have 2CPUs, 4GB RAM, 70GB disk (except where noted), 3 Network interfaces (except where noted). There are 5 different networks type setup on these interfaces:

1. PXE Network
2. Public Network
3. Private Network
4. Management Network
5. Storage Network

Environment setup reference docs:

<https://docs.mirantis.com/openstack/fuel/fuel-7.0/quickstart-guide.html#installing-mirantis-openstack-using-the-mirantis-virtualbox-scripts>

The Storage network shall be connected to the Unity iSCSI ports to provide Unity data path for Compute nodes and Controller nodes.

## Product compatibility matrix

|  |  |  |
| --- | --- | --- |
| Unity Plugin version | Compatible Fuel version | OpenStack Version |
| 1.0.0 | 7.0 | Kilo |
| 8.0 | Liberty |
| 9.0 | Mitaka |

# Type of testing

*Testing of the plugin shall include installation and functional tests.*

## Installation testing

|  |  |
| --- | --- |
| Test Case ID | *Install and remove plugin* |
| Description | *Prove that the plugin installs and uninstalls properly* |
| Prerequisites | *A properly configured Fuel master* |
| Steps | 1. *Run system tests #1, 4 and 5 (see ‘system test’ below)* |
| Expected Result | *The plugin should install and uninstall properly.* |

## Functional testing

|  |  |
| --- | --- |
| Test Case ID | *Functional test of proper cinder.conf output* |
| Description | *Prove that the plugin properly sets up the cinder.conf file on deployment and removal of nodes.* |
| Prerequisites | *A properly configured Fuel master and 5 test nodes.* |
| Steps | 1. *Run systems tests #1, 2, and 3 (see ‘system test’ below)* |
| Expected Result | *The plugin shall accept the necessary inputs and configure the necessary variables in the cinder.conf file on all Controller nodes in the environment. The plugin shall properly accommodate addition and removal of controller nodes.* |

|  |  |
| --- | --- |
| Test Case ID | *Functional test of volume* |
| Description | *Prove that cinder operation can be performed against Unity array* |
| Prerequisites | *A properly configured Fuel master and 5 test nodes.* |
| Steps | 1. Create volume type with name “unity-type”  2. Set extra-specs “volume\_backend\_name=emc-unity” for the created type  3. create volume with type “unity-type”  4. Verify the volume is in available state, and check it on Unity, make sure it’s created on Unity instead of other storage backends.  5. Create VM instance in nova  6. attach the volume to the instance, make sure the volume is in “in-use” state |
| Expected Result | *The plugin shall accept the necessary inputs and configure the necessary variables in the cinder.conf file on all Controller nodes in the environment. The plugin shall properly accommodate addition and removal of controller nodes.* |

## System testing

## Install plugin and deploy environment

|  |  |
| --- | --- |
| Test Case ID | install\_plugin\_deploy\_env |
| Steps | 1. Upload plugin to the master node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel UI 5. Add 3 nodes with Controller role and 1 node with Compute and another role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Create a volume within OpenStack, confirm volume is on Unity |
| Expected Result | *Plugin is installed successfully, cluster is created, network* verification and all plugin services are enabled and work as expected. |

## Modifying env with enabled plugin (removing/adding controller nodes)

|  |  |
| --- | --- |
| Test Case ID | modify\_env\_with\_plugin\_remove\_add\_controller |
| Steps | 1. Upload plugin to the master node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 1 node with Compute and another role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Create a volume within OpenStack, confirm volume is on Unity 10. Remove 1 nodes with Controller role   /\*Remove node, where plugin’s services available, to ensure that according to ha mode all plugins resources will be replaced and available on another live node and continue to work as expected\*/   1. Re-deploy cluster 2. Create a volume within OpenStack, confirm volume is on Unity 3. Add 1 new node with Controller role 4. Re-deploy cluster 5. Create a volume within OpenStack, confirm volume is on Unity |
| Expected Result | *Plugin is installed successfully, cluster is created, network* verification and all plugin services are enabled and work as expected. |

## Modifying env with enabled plugin (removing/adding compute node)

|  |  |
| --- | --- |
| Test Case ID | modify\_env\_with\_plugin\_remove\_add\_compute |
| Steps | 1. Upload plugin to the master node 2. Install plugin 3. Ensure that plugin is installed successfully using cli 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 2 nodes with compute roles and 1 another role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Create a volume within OpenStack, confirm volume is on Unity Run OSTF 10. Remove 1 compute node 11. Re-deploy cluster 12. Create a volume within OpenStack, confirm volume is on Unity Add 1 compute node 13. Re-deploy cluster 14. Create a volume within OpenStack, confirm volume is on Unity |
| Expected Result | *Plugin is installed successfully, cluster is created, network* verification and all plugin services are enabled and work as expected. |

## Uninstall of plugin with deployed environment

|  |  |
| --- | --- |
| Test Case ID | uninstall\_plugin\_with\_deployed\_env |
| Steps | 1. Install plugin 2. Deploy environment with enabled plugin functionality 3. Run ostf 4. Try to delete plugin and ensure that present in cli alert: "400 Client Error: Bad Request (Can't delete plugin which is enabled for some environment.)" 5. Remove environment 6. Remove plugin 7. Check that it was successfully removed |
| Expected Result | *Plugin is not uninstalled and an error is returned indicating non-removal due to being in use.* |

## Uninstall of plugin

|  |  |
| --- | --- |
| Test Case ID | uninstall\_plugin |
| Steps | 1. Install plugin 2. Check that it was installed successfully 3. Remove plugin 4. Check that it was successfully removed |
| Expected Result | *Plugin is installed and then is removed successfully* |